

1 ELECTRONIC MAIL SOFTWARE WITH MODULAR INTEGRATED AUTHORING/READING
2 SOFTWARE COMPONENTS INCLUDING METHODS AND APPARATUS FOR
3 CONTROLLING THE INTERACTIVITY BETWEEN MAIL AUTHORS AND RECIPIENTS

4 *2us*
a'
sub
ch
5 This application is a continuation-in-part of application
6 serial number 09/209,162 filed December 10, 1998, and serial
7 number --/---,--- filed ----, 2000 [BAK-006] the complete
8 disclosure of which is hereby incorporated by reference herein.

9
10 BACKGROUND OF THE INVENTION

11
12 1. Field of the Invention

13 The invention relates to an electronic mail program. More
14 particularly, the invention relates to an electronic mail program
15 having modular integral authoring/reading applications whereby
16 documents created with the modular integral authoring/reading
17 applications are seamlessly sent and received by the mail program
18 and which provides different kinds of interactivity with and
19 different kinds of access to electronic mail messages depending on
20 user types or roles.

21
22 2. State of the Art

23 In recent years electronic mail ("email") has become widely
24 used in business, education, and in personal communications. One

1 of the features of electronic mail which is most convenient,
2 particularly in business and in education, is the ability to
3 attach a binary computer file to an email message. This feature
4 enables email correspondents to rapidly share word processing
5 documents, database documents, spreadsheet documents, multimedia
6 documents, or virtually any kind of binary file created by a
7 computer. There are, however, some serious limitations and
8 inconveniences associated with attaching a binary file to an email
9 message.

1 The original Internet mail system as defined in 1982 with RFC
2 (Request for Comments) 821 and 822 had a number of important
3 limitations. In particular, the system was not designed to carry
4 large quantities of arbitrary data in an email message. In fact,
5 the 1982 SMTP (Simple Mail Transport Protocol) standard required
6 that an email message consist of a single message containing only
7 ASCII characters in lines of 1000 characters (blocks of 32k) or
8 less. Moreover, some implementations of SMTP or other mail
9 transport systems (such as UUCP) restricted message lengths to
10 some allowed maximum number of bytes. Lengthy messages passing
11 through a mail gateway using one of these implementations were
12 likely to be truncated.

1 The ability to send large quantities of binary data through
2 the Internet electronic mail system was made possible with the
3 MIME (Multipurpose Internet Mail Extensions) standard for Internet
4 messages. The original MIME standard was published as an Internet
5 Request For Comments document (RFC 1341) and approved in June of
6 1992. (See Internet RFCs 2045, 2046, and 2047 for the latest MIME
7 standards documents.) The MIME standard describes how an email
8 message should be formatted in order to be considered MIME
9 compliant. MIME defines a set of message header fields and a set
10 of message encoding standards that are designed to overcome the
11 limitations of RFC 822 message formats and still be transportable
12 through any of the numerous legacy mail transport systems in use
13 on the Internet. MIME message header fields extend those defined
14 in RFC 822 and describe the content and encoding type of the email
15 message. Encoding schemes allowed in the MIME standard include
16 "quoted-printable", and "base64". In addition, three unencoded
17 data types are allowed. These are labeled "8bit", "7bit", or
18 "binary".

19
20 If the sender and the receiver of the email message with the
21 attached binary file are using the same brand and version of email
22 program and both programs are configured in substantially the same
23 way, the receiver's email program should automatically apply the
24 appropriate decoding to the attached binary file and produce a

1 file which is identical to the file which was attached to the
2 email by the sender. However, if the sender and receiver are
3 using different email programs, the recipient may receive a file
4 which must be decoded by the recipient using a separate decoding
5 program.

6
7 Even after the file is properly received and decoded, it is
8 often difficult for the receiver of the file to open the file.
9 The receiver of the file might expect that "clicking" on the file
10 icon will open the file. However, clicking on the file icon will
11 often not open the file. It may result in an error message like
12 "application not found" or, worse, it may result in the file being
13 opened by an inappropriate application thereby displaying
14 "gibberish". The receiver of the file must have a program capable
15 of reading (opening) the file. For example, if one attaches a
16 spreadsheet file to an email message, the receiver of the file
17 must have a spreadsheet program in order to open the file.
18 Technically, it is not necessary that the receiver of the file
19 have the same brand program as that which created the file.
20 However, opening a file with a program which did not create it,
21 though possible, can be very inconvenient. The receiver of the
22 file must know what kind of file is attached to the email message,
23 must know what program on their computer is capable of reading
24 that type of file, must launch the program, must open the file

1 from within the program, and wait while the program translates the
2 file.

3

4 The limitations of Internet electronic mail can become even
5 more frustrating if the sender and recipient are not using the
6 same operating system (OS). Some mail attachment encoding schemes
7 (and file compression schemes) are OS-dependent and it is possible
8 that an email recipient could receive a file which is impossible
9 to decode (or decompress).

10

11 These limitations in electronic mail have discouraged many
12 people, particularly non-sophisticated computer users, from
13 attaching files to electronic mail messages. In fact, for some
14 novice users, the task of launching one application to create a
15 document, saving the document, launching a separate email
16 application to create an email message, and then locating the
17 saved document for attachment to an email message is daunting
18 enough to discourage them. In addition, novice users often
19 complain that after "downloading" a file attached to an email
20 message they cannot find the file on their hard disk.

21

22 Another interesting aspect of electronic mail is that it is
23 now widely used in electronic commerce, but only in a very limited
24 way. Electronic mail is used by vendors to advertise goods and

1 such electronic mail may typically include a hyperlink which, if
2 clicked on, will cause the mail recipient's computer to launch a
3 web browser and connect to the vendor's website where the goods
4 may be purchased. Electronic mail is also used by online vendors
5 to provide customer support by answering questions sent to the
6 vendor via electronic mail and to confirm online purchases by
7 sending electronic mail to the purchaser.

8
9 My previously incorporated parent application discloses
10 electronic mail software which includes a main email component and
11 a number of installable components. The installable components
12 include authoring/reading components for creating/reading
13 different kinds of documents and mailbox components for listing
14 different kinds of messages or for listing messages in different
15 styles. The main email component provides an underlying graphical
16 user interface for functions directly associated with the storage
17 and transfer of electronic mail messages, and also handles all
18 data bundling and unbundling required to transform a message
19 created by an authoring component into a MIME compliant message.
20 The authoring/reading components act like applications embedded
21 within the email program and allow specific types of documents
22 such as spreadsheets, graphics, databases, etc. to be created from
23 within the email program and emailed directly. The
24 authoring/reading components also allow received documents to be

1 read without the difficulties traditionally associated with
2 attaching binary files to an email letter. The authoring
3 components of the invention pass data to the main email component
4 which packages the data as a MIME compliant message. When the
5 message is received, the main email component concatenates (as
6 needed) and decodes the MIME message and sends the data to the
7 authoring/reading component associated with the MIME type. The
8 electronic mail software also includes modular integrated
9 authoring/reading software wherein the functionality of the
10 authoring/reading software is controlled by the "role" of the user
11 when participating in an exchange of messages. One example of
12 "roles" given in the parent application was that of teacher and
13 student. Another example was that of a puzzle encoder and a
14 puzzle decoder.

15
16 It is believed that the enhanced electronic mail software
17 disclosed in my previously incorporated parent application may be
18 even further enhanced to provide more functionality regarding the
19 use of "roles" in the contexts of electronic commerce, healthcare,
20 business, and law, as well as in education.

21
22 Recently, the email client OUTLOOK by MICROSOFT has received
23 much attention because of its ability to execute Visual Basic
24 Scripts (VBS) attached to or embedded in the body of an email

1 message. While this ability has the potential of achieving some
2 of the goals of the present invention, it has one major
3 disadvantage. The recent interest in MICROSOFT OUTLOOK and VBS
4 was spurred by the outbreak of several extremely damaging
5 "viruses" or "worms". These Visual Basic Scripts attached to
6 email destroyed files on the email recipient's hard drive and
7 spread by sending copies to all of the email addresses in the
8 recipients' address books. At the present time, the only sure way
9 to prevent damage by malicious Visual Basic Scripts is to alter
10 the "preferences" of OUTLOOK so that the ability to execute Visual
11 Basic Scripts is disabled.

SUMMARY OF THE INVENTION

12
13
14
15 It is therefore an object of the invention to provide an
16 electronic mail program which includes integrated
17 authoring/reading software whereby different kinds of documents
18 may be created, sent, received, and opened by electronic mail in a
19 seamless manner.

20
21 It is another object of the invention to provide an
22 electronic mail program which includes modular integrated
23 authoring/reading software whereby different kinds of documents
24 may be created and sent by email in a seamless manner.

1 It is a further object of the invention to provide an
2 electronic mail program which includes modular integrated
3 authoring/reading software wherein the functionality of the
4 authoring/reading software is controlled by the "role" of the user
5 when participating in an exchange of messages.
6

7 It is an additional object of the invention to provide an
8 electronic mail program which includes modular integrated
9 authoring/reading software whereby the role of the user includes
10 such roles as customer, vendor, database, service provider,
11 technical support, teacher, student, attorney, client, doctor,
12 patient, and organization members having different security
13 clearances.
14

15 It is another object of the invention to provide an
16 electronic mail program whereby the role of the message recipient
17 is automatically generated by the authoring/reading component and
18 encoded in the outgoing message.
19

20 It is still another object of the invention to provide an
21 electronic mail program which includes installable
22 authoring/reading components which communicate with the email
23 program through a bidirectional application programming interface
24 (API).

1 In accord with these objects which will be discussed in
2 detail below, the electronic mail software according to one
3 embodiment of the present invention includes a main email
4 component and a number of installable components which communicate
5 bidirectionally with the email component through an application
6 programming interface (API). The installable components include
7 authoring/reading components as well as at least one mailbox
8 browser/editor component. The main email component provides an
9 underlying graphical user interface (GUI) for functions directly
10 associated with the storage and transfer of electronic mail
11 messages. In particular, the main email component provides menu
12 items which allow the user to SEND, READ, REPLY, FORWARD, DELETE,
13 SAVE, and PRINT, for example. The main email program also handles
14 all data bundling and unbundling that may be required to transform
15 a message created by an authoring component into a fully MIME
16 compliant message. In addition, the main email component includes
17 "hooks" (an application programming interface or API) for the
18 attachment of the installable components. The authoring/reading
19 components each provide functionality which is particular to the
20 type of document the component is designed to create/display.

21
22 According to the invention, some modular components have
23 assigned "roles" whereby senders and recipients of certain email
24 documents are provided different kinds of access to the documents.

1 For example, in the case of customer/vendor components, the vendor
2 component allows a vendor to create an order form which appears in
3 an email message read by the customer. The customer component
4 allows the customer to fill in the form and mail back the filled-
5 in data to the vendor or the vendor's database. A database
6 component automatically reads the order data from the customer
7 email and transfers this data to billing and fulfillment software.
8 Other modular components having assigned roles include
9 bidder/auctioneer, doctor/patient, attorney/client, etc.

10
11 The authoring/reading components interface with the main
12 email component via designated "MIME types". The MIME data
13 standard allows developers to define MIME types using the label
14 "x-"`<string>` in the data header where `<string>` is any ASCII string
15 excluding space, control characters and other special characters.
16 The MIME data standard also allows developers to define MIME
17 subtypes using an extension to the label that appends the
18 character "/" and a string to the MIME type. For example a Kidcode
19 Rebus message might be identified with the content-type header set
20 as "Content-Type: x-KidCode/Rebus". Further, the MIME standard
21 allows developers to include parameters within the contents of the
22 "Content-Type" header field. For example, the KidCode Rebus
23 message might be associated with a particular feature such as the
24 school grade level associated with this type of Rebus message,

1 e.g. "Content-Type: x-KidCode/Rebus grade = 1". The authoring
2 components of the invention pass data to the main email component
3 which packages the data as a MIME compliant message with the label
4 "x-"<application> in the message header, where <application>
5 identifies the authoring/reading component which created or can
6 display the message. When the message is received, the main email
7 component concatenates and decodes the MIME message, reads the
8 MIME type, sends the data to the component associated with the
9 MIME type, and waits for a user event or a callback from the
10 component. This bidirectional communication between the main
11 email component and the authoring/reading components provides a
12 totally seamless operation wherein the user may send and receive
13 complex documents without any knowledge of attaching files,
14 downloading, decoding, etc. Moreover, enhanced functionality is
15 achieved through installable components. Only after the
16 appropriate component has been installed can an "active" email
17 message call upon the component to execute.

18
19 Additional objects and advantages of the invention will
20 become apparent to those skilled in the art upon reference to the
21 detailed description taken in conjunction with the provided
22 figures.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a very high level schematic diagram of the architecture of the email software of the invention;

Figure 2 is a somewhat lower level schematic diagram of the architecture of the email software of the invention;

Figure 3 is a screen shot of an email message created with a vendor authoring component and displayed by a generic authoring/reading component which can control the generation of new messages for other authoring/reading components;

Figure 4 is a screen shot of an uncompleted email message template generated by a customer reading/authoring component in response to the email message shown in Figure 3;

Figure 5 is a screen shot of the completed template of Figure 4;

Figure 6 is a screen shot of an interactive template created by an auctioneer authoring component and displayed by a bidder/seller authoring/reading component displayed in the bidder role;

1 Figure 6a is a screen shot of an interactive template created
2 by an auctioneer authoring component and displayed by a
3 bidder/seller authoring/reading component displayed in the seller
4 role;

5
6 Figure 7 is a screen shot of a doctor reading/authoring
7 component;

8
9 Figure 8 is a screen shot of a patient reading/authoring
10 component;

11
12 Figure 9 is a state diagram illustrating how roles are
13 automatically assigned; and

14
15 Figure 10 is a state diagram illustrating the changes in role
16 in an authoring/reading component.

17
18 DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

19
20 An exemplary embodiment of the architecture of the invention
21 is described as follows with reference to an email client which
22 will be distributed under the brand name UMBANET. Referring now
23 to Figure 1, the email client includes a main client 10, having a
24 main client API 12, and one or more authoring/reading components

1 14, each having a component API 16. As described in the
2 previously incorporated parent applications, the authoring/reading
3 components each include a message handler 18 having an API and
4 preferably an installable mailbox handler 20 having an API.
5 Moreover, as described below with reference to Figure 2,
6 installable components may be made up of either message handlers,
7 mailbox handlers, or both.

8
9 Referring now to Figure 2, the main email client 10 includes
10 functionality for message transport 22, message encoding/decoding
11 24, message storage and retrieval 26, basic GUI to mail services
12 28, user administration 30, component installation 32, and special
13 services 34. As mentioned above, the main email client 10 also
14 includes an API 12 for communicating with authoring/reading
15 components 14a, 14b, 14c, each of which have an API 16a, 16b, 16c
16 for communicating with the main email client 10. In addition, as
17 mentioned above, a separate installable mailbox browser/editor 14d
18 may be provided, having its own API 16d for communication with the
19 API 12 of the main email client 10. An example of a specialized
20 mailbox browser/editor component is disclosed in previously
21 incorporated parent application [BAK-006].

22
23 The main email client 10 performs all of the usual functions
24 of an electronic mail client. These include message transport,

1 message storage and retrieval, message encoding and decoding, user
2 administration (including user preferences). In addition, the
3 main email client includes unique functionality required to enable
4 installable components and special server-based services.

5
6 The message transport functionality 22 of the main email
7 client 10 handles all functions traditionally associated with
8 sending (SMTP) and receiving (POP, IMAP) email messages. This
9 includes finding and verifying network addresses, and sending and
10 receiving mail messages to other servers on a network.

11
12 The main email client 10 handles all data bundling and
13 unbundling that may be required to transform the message data used
14 by a message authoring component into a fully MIME compliant
15 message type via the message encoding/decoding module 24. This
16 way each message authoring component handles data in a format most
17 convenient to it and all MIME parsing and details associated with
18 protocol compliance are centralized in the main email client 10.
19 The only requirement for the message data passed between a message
20 authoring/reading component and the main email client is that the
21 message body data be formatted either as an ASCII string or in a
22 binhex format.

23

1 The main mail client 10 maintains its user mailbox files
2 locally. The storage and retrieval module 26 implements all
3 functionality required for reading and writing messages and
4 mailboxes to/from permanent storage.

5
6 The main email client 10 also includes a GUI 28 for user
7 control of those functions that are directly associated with
8 storage, display, and transfer of electronic mail messages and
9 mailboxes. In particular, the main email client interface
10 includes buttons and/or menu items that allow a user to send a
11 message, reply to a message, open a message or a mailbox,
12 delete/trash messages or mailboxes, save a message to an
13 alternative mailbox, and print a message. According to the
14 presently preferred embodiment, the main email client user
15 interface operates simultaneously with the user interface of an
16 authoring/reading component in the sense that both are active on
17 the screen at the same time and present the "look and feel" of a
18 single application. For example, Figures 6-8 show authoring
19 reading components for an auction bidder, a doctor, and a patient
20 respectively. As will be explained in more detail below with
21 reference to Figures 6-8, each of the authoring/reading components
22 has its own interface which permits the user to interact with the
23 component in a manner particular to the component. As seen in
24 Figures 6-8 the authoring/reading component interface is generally

19

1 module includes administrative functions and a user interface to
2 add and modify information for more than one user. The module
3 maintains a data structure consisting of all relevant user
4 information for each user in the system. The module also manages
5 files that are required for permanent storage of user information,
6 if necessary.

7
8 The component installer module 32 handles the installation of
9 installable components (authoring/reading, mailbox, etc.) and
10 maintains a data structure of component information for each
11 installable component. When a new message handling component is
12 installed into the mail client, the following information is made
13 accessible to the main mail client via the component installer:
14 MIME type of message handled by the component, component name and
15 type, location and file name of program executable code for the
16 component, location(s) of icon image data for the component, and
17 description of the resources that need to be installed into the
18 main client, e.g. buttons, menu items etc.

19
20 The implementation of the component installer is dependent
21 upon the programming environment within which the system is built
22 and meant to operate. In order for installable components to
23 operate properly, it is necessary for the main client program to
24 be able to execute the new component's program code and make

1 function calls to that code. It is not sufficient to execute the
2 code as a completely separate program module because function
3 calls and data must be passed between the main program and the
4 component. For example, in a Windows environment, an installable
5 component might be made up of one or more DLL's (dynamic link
6 libraries) and one or more resource or ini files. The email
7 clients described in the previously incorporated parent
8 applications were implemented with MACROMEDIA DIRECTOR using
9 protected movies and an MIAW (movie in a window) invocation
10 architecture. A Java implementation would use loadable Java
11 classes to implement installable components.

12
13 The component installer should also be capable of editing
14 data structures in the main program that manage the display of
15 relevant user interface controls. For example, the component
16 installer for the KIDCODE® client (described in previously
17 incorporated serial number 09/209,162) adds a new button with the
18 component icon to the main screen as well as making an icon
19 available to the mailbox display component. It is also generally
20 desirable to add entries for the new component to menus in the
21 main client program.

22
23 An important part of the design of the component installer is
24 a specification document that describes the format and type of

1 information that must be provided by each installable component in
2 order to have that information installed into the main client
3 program.

4
5 The special services module 34 allows the UMBANET mail client
6 to communicate with UMBANET servers on the Internet in order to
7 provide specialized services such as component downloads, message
8 tracking and accounting, recipient mail capability verification,
9 PKI security and certification, etc. The UMBANET client uses
10 TCP/IP and a proprietary protocol to communicate with UMBANET
11 servers. Once a TCP/IP connection has been established with an
12 UMBANET server, the client registers with the server. Thereafter,
13 the client and server exchange information using the UMBANET
14 protocol.

15
16 As mentioned above, the mailbox and the authoring/reading
17 components are preferably "installable". However, it is possible
18 to create an email client having most of the desired functionality
19 of the invention with fixed components hard coded. In either
20 case, the invention prescribes that mailbox components are used to
21 display, edit, and browse mailboxes. Different kinds of users and
22 different types of messaging applications (e.g. fax, traditional
23 email, internet voice) may require very different displays and
24 functionality from a traditional mailbox viewer/editor.

1 Installable mailbox components make it possible to upgrade, select
2 from multiple viewing formats, utilize different mailbox
3 viewer/editors for different users, and in general increase the
4 range of functionality that can be achieved within one basic
5 messaging application program.

6
7 The message authoring/display (message handler) components
8 make it possible to handle an unlimited number of message types.
9 Each message handler component is designed to deal with a specific
10 MIME type of message. The MIME data standard has been designed so
11 that application developers can define new MIME types as needed by
12 labeling these with the "x-"<string> prefix. A message handler
13 component can be any program that defines a message MIME type of
14 data that it handles, implements the callback API functions
15 described below, and matches the requirements imposed by the
16 component installer module. API callback functions allow the main
17 email client to obtain information about the message handlers and
18 allow the message handlers to respond to standard mail commands
19 such as Send or Reply, that have been issued by a user through the
20 main email interface.

21
22 As described in more detail below with reference to Figures
23 3-8, some of the types of message handler components contemplated
24 by the present invention are those which can be thought of as

1 creating "roles" among senders and recipients of mail messages.
2 As will be further elaborated, either the message handler or the
3 message created by it can establish the "roles" of the recipients
4 and the authors such that recipients and authors are each
5 permitted to interact with the message in certain different
6 predefined ways.

7
8 Before describing the specific message handlers of the
9 instant invention, it is useful to first fully describe the
10 preferred APIs used to implement the main email component, the
11 mailbox components, and the message handlers.

12 Main Email API Functions

13
14 These functions are called by the installable components to
15 access services provided in the main email program.

16 17 **emh_getUserMailbox** (dt_MailBoxName)

18 This returns a mailbox data structure for the current user
19 and mailbox name. This function is normally called by a mailbox
20 handling component. Mailbox handling components may use temporary
21 files to hold mailbox contents but they preferably never access
22 the users mailbox files. All access to these files should be
23 obtained through the main email program. The input to this
24 function is (dt_MailBoxName) and the output is dt_MailboxData.

1 **emh_getUserData** (dt_UserName)

2 This returns a data structure with user information (in a
3 multi-user client). The main email program maintains all user
4 information and handles user administration functions. The main
5 program also communicates with external mail servers which may
6 contain other user information. The input to this function is
7 (dt_UserName) and the output is dt_UserData.

8

9 **emh_continue**

10 This function is used by installable components to explicitly
11 pass control back to the main email program. This function is
12 necessary for the present implementation which uses MACROMEDIA
13 DIRECTOR LINGO code. It is not a requirement for alternative
14 implementations. The function has the argument
15 (dt_ComponentType).

16

17 **emh_killComponent**

18 This function is used by an installable component to inform
19 the main email program that it is preparing to terminate. This
20 allows the main program to free any memory and/or data structures
21 that have been allocated to the component.

22

23

1 emh_passMessage

2 This function is used primarily by mailbox components to pass
3 a message to the main program so that it can be displayed by the
4 appropriate message handling component. The main program takes
5 the message arguments (em_MailData) or (em_MessageAction). The
6 former looks up the Mimetype of the message, and invokes the
7 appropriate message handler to display the message. The latter is
8 used with #open, #print, #send,...etc.

10 emh_getMessage

11 This function returns the message (em_MailData) with Number
12 MessageNumber from the MailboxName of the current user. It can be
13 used by installable components to retrieve specific messages from
14 the user's mailboxes. If this is called with the messageNumber
15 set to 0, the main email assumes the typeOrBoxName argument is a
16 mimetype and returns a new message structure. Message handling
17 components should call emh_getMessage with the number set to 0 and
18 the mimetype whenever a new message is started. Normally this
19 should be done whenever an active message is trashed.

20

21

1 emh_getRegisteredUsers

2 This returns a list of usernames (dt_RegisteredUsers) for the
3 users that are registered with the email client, i.e. that have
4 been added as users by the user administration part of the main
5 email program. This is the same list of users that appears in the
6 logon listbox when the program is started up. It may be used by
7 installable components to create listboxes for filling address
8 fields in messages or for checking whether a particular address is
9 external to the system. This function may be used with an
10 argument that specifies a MIME type. When so used, the function
11 will return a list of users who are authorized to receive (or
12 capable of displaying) messages of the specified MIME type. The
13 indications of the authorizations or capabilities of users may be
14 stored on a local server, on a remote server (via the Internet),
15 or may be stored locally on a user's computer.

17 emh_sendMessage

18 The main email client sends a message with the argument
19 (em_MailData) by either forwarding it to an external mail server
20 or, if it is a registered email client user, writing the message
21 to the user's incoming mail mailbox.

22

23

1 emh_saveMessage

2 The main email client saves a message with the argument
3 (em_MailData) for the currently logged on user by writing the
4 message to the user's "notes in progress" mail mailbox.

6 emh_disableButton

7 It is recommended that this function be used carefully.
8 Normally, the main email program controls the state of all the
9 buttons available to users to access message handling of the main
10 program. This function can be used to request that the main email
11 program disable the button specified by the argument, ButtonName.
12 If the button is disabled, whether it was already disabled or is
13 disabled as a result of this function call, this function will
14 return TRUE, otherwise it will return FALSE. The calling
15 component should check whether the function call succeeded and
16 proceed accordingly. The ButtonNames are #reply, #send, #print,
17 #trash, etc.

19 emh_enableButton

20 It is recommended that this function be used carefully.
21 Normally, the main email program controls the state of all the
22 buttons available to users to access message handling of the main
23 program. This function can be used to request that the main email

1 program enable the button specified by the argument, ButtonName.
2 If the button is enabled, whether it was already disabled or is
3 disabled as a result of THIS function call, this function will
4 return TRUE, otherwise it will return FALSE. The calling component
5 should check whether the function call succeeded and proceed
6 accordingly.

7

8 API Functions For Use With All Component Types

9

10 **emc_startMeUp**

11 This function is used by the main email program to tell an
12 installable component to start. This function will execute prior
13 to initialization of the component's data structures, which should
14 only be initialized after the component receives the
15 emc_initWindow call from the main email program. This function is
16 necessary for the MACROMEDIA DIRECTOR LINGO code implementation
17 and may not be needed for other implementations.

18

19 **emc_initWindow**

20 This function is used by the main email program to tell an
21 installable component to initialize its data structures and
22 prepare its graphical display. The component is passed the
23 username of the current user. If the component requires

1 additional user information in order to initialize, it can call
2 emh_getUserInfo within its implementation of this function.

4 **emc_closeWindow**

5 This function is used by the main email program to tell an
6 installable component to free all memory that it has used, close
7 its window, and shut down.

9 **emc_getComponentInfo**

10 This function is used by the main email program to get
11 required information such as componentName, componentID, etc. from
12 an installable component. This function should not be called
13 until the component window is fully open.

15 API Functions For Use With Mailbox Handler Components

17 **mbx_getMessageNumbers**

18 This function is used by the main email program to get the
19 message number of the currently selected message in the mailbox
20 browser. If no message is selected, the list should be empty.

22 **mbx_getMessage**

23 This function is used by the main email program to get the
24 message data structure of the message with em_MessageNumber from

1 the mailbox currently displayed in the mailbox browser. If the
2 function fails, e.g. if there is no message with the given message
3 number, the function returns an empty list.

5 **mbx_trashMessages**

6 This function is used by the main email program to tell the
7 mailbox component that it has received a trash event (e.g. user
8 pressed trash button). The component should identify which
9 messages have been indicated by the user and return these messages
10 in a list data structure. If no messages have been indicated by
11 the user or if the component wishes to cancel the operation it
12 should return an empty list. When it receives this call, normally
13 the mailbox component will update its display and its data
14 structures to delete messages that have been indicated by the
15 user.

17 **mbx_openMailbox**

18 This function is used by the main email program to tell the
19 mailbox component to display the mailbox passed in the argument
20 (dt_Mailbox).

Functions For Use With Message Handler Components

msh_sendMessage

This function is used by the main email program to tell a message handling component to pass back a fully completed message data structure so that it can be sent to the recipient specified in the message's address field. The message handling component should update its display as appropriate for a message that has been sent. It should also change its state to display mode because a message that has already been sent should not be editable. If the function fails, e.g. if a fully completed message cannot be constructed (for example, if the user has not specified a message recipient), the function returns an empty list.

The message handling component will normally control all dialogs with a user that pertain to a message under construction. For example, to alert the user that a message recipient must be specified. However, if the message handling component fails to pass back a properly formatted, completed message data structure, (or an empty list acknowledging failure), the main email program will detect the error and alert the user about the field or fields that have not been specified.

1 msh_openMessage

2 This function is used by the main email program to pass a
3 message data structure to a message handling component so that it
4 can be displayed. The message handling component should display
5 the message in the specified mode (e.g. either #author or
6 #display). If the em_Mode argument is #display the message should
7 not be editable. Otherwise the message should be opened so that
8 it can be edited. If the function fails, e.g. if an error is
9 detected in the message body, the message handler returns FALSE,
10 otherwise the message handler returns TRUE.

12 msh_replyMessage

13 This message is used by the main email program to inform a
14 message handling component to display the currently active message
15 for editing as a reply. In order to reply, the message handling
16 component will generally create a new message with the mode set to
17 #author. If the function fails, e.g. if an error is detected in
18 the message body, the message handler returns FALSE, otherwise the
19 message handler returns the messageBody which may have been
20 modified.

1 msh_clearMessage

2 This function is used by the main email program to inform a
3 message handling component that the current message should be
4 cleared from the display and from the message handling component's
5 data structures. This function is used, for example, when the
6 user indicates they want to trash the current message by clicking
7 on the "trash" button in the main email program GUI. If the
8 function fails, the message handler returns FALSE. Otherwise the
9 message handler returns TRUE.

11 msh_saveMessage

12 This function is invoked when the main email program receives
13 a save event, e.g. the user presses the "Save" button. The
14 message handler should pass the messageBody back to the main email
15 program to be saved. In general, the message handler should
16 maintain its state when this function is called because the user
17 may want to continue editing the message.

19 msh_printMessage

20 This function is used by the main email program to inform a
21 message handling component that a message should be printed. This
22 function is used, for example, when the user indicates they want
23 to print the current message by clicking on the "print" button in
24 the main email program GUI. When the argument, em_mailData, is an

1 empty list, the message handler component should print the
2 currently active message. Otherwise the message handler component
3 should print the message argument. Normally, if the message
4 handler component has been fully initialized and is displayed in a
5 window, the main email program will call this function with an
6 empty list for an argument. The function may also be used by the
7 main email program to have a message handler print a message even
8 though the message handler component has not been fully
9 initialized and displayed in a window. For example, this will
10 occur if an active mailbox component receives a print request from
11 the main email program for a message that has been selected in the
12 mailbox browser. In this case, the main email program will send a
13 request to the appropriate message handler component to print the
14 message without fully starting it up and initializing its window.
15 Therefore the message handler should implement the
16 msh_printMessage function so that the following sequence of
17 function calls succeeds emc_startMeUp, msh_printMessage(message).
18 If the function fails, the message handler returns FALSE.
19 Otherwise the message handler returns TRUE.

21 **msh_trashMessage**

22 This function is used by the main email program to tell a
23 message handler component that it has received a trash event (e.g.
24 user pressed trash button). The component should do whatever is

1 appropriate and return its currently open message data structure.
2 A component may want to display a dialog box to verify the trash
3 operation before proceeding. (e.g. emh_alertUserToTrash) If the
4 component wishes to cancel the operation, it should return an
5 empty list. Normally a message handling component will update its
6 display and its data structures to clear the trashed message from
7 its display.

8
9 From the foregoing, those skilled in the art will appreciate
10 that message handling components may be created for different user
11 roles and/or messages may be assigned certain role information
12 which controls how the recipient(s) of the message interacts with
13 the message. Moreover, certain message handling components can
14 provide different levels of authoring and different levels of
15 readership based on "roles."

16
17 Turning now to Figures 3-6, one implementation of a message
18 handling component is that of a "customer" component (or a
19 reading/authoring component which is responsive to a "customer
20 message role). Figure 3 illustrates an email message 40 which was
21 created by a "vendor" component (or an authoring/reading component
22 in a "vendor" role). Figure 3 shows the message as it appears on
23 the recipient's computer screen. As shown the message appears to
24 be a simple plain text email message with a hot link 42 to the

1 vendor's web page (the underlined sentence "Click here to Go to
2 Our Web Page for more Details."). However, the second hot link 44
3 in this message (the sentence "Click here to Automatically Order
4 the Product by Email") invokes special functionality of the
5 reading/authoring component installed in the recipient's email
6 program. When the user clicks on this second hotlink, the
7 "customer" message handling component causes a new outgoing email
8 message 46 to be created as shown in Figure 4.

9
10 As shown in Figure 4, the new mail message is pre-addressed
11 and the body contains information relating to the order which will
12 be placed when the new message is sent. The body of the message
13 is actually a template which was created from information
14 contained (though hidden from display) in the email message 40
15 received from the vendor. This information was not displayed in
16 the original message shown in Figure 3, but was made known to the
17 message handling component for use in creating the template 46
18 shown in Figure 4 if the user clicked on the second hot link 44 in
19 Figure 3. The customer role assigned to this outgoing email
20 message 46 allows the user to make only limited additions to this
21 template. Specifically, the user may enter a number of items in
22 the blank 48 and a choose a size using the "radio buttons" 50,
23 then click on the send mail icon 52. This template is interactive
24 and controlled by the message handling component so that as soon

1 as the user selects a number to order, the purchase price is
2 displayed in the field 54 as shown in Figure 5. In order to
3 minimize the amount of input required of the customer, the
4 purchase price is preferably charged to an account already set-up
5 with the vendor. This mode of operation also maximizes the
6 security of the transaction.

7
8 As mentioned above, the user can click on the send mail icon
9 52 once the form is filled in. Preferably, the send mail icon 52
10 will remain "grayed out" until the user completes all of the
11 information required by the form. If the user changes his/her
12 mind about placing the order, s/he may simply close this window
13 without sending the message. If the user clicks on the send icon
14 52, the mail is sent back to the vendor, preferable with an
15 assigned role of "order fulfillment", for example. The message
16 may also be preferably addressed to an automatic mail handler
17 which will automatically extract information from the order and
18 from the customer database and process the order.

19
20 Figure 3 shows what looks like a standard HTML message but is
21 used to open a customer reading/authoring component displayed in
22 Figures 4 and 5. The screen shot of Figure 3 doesn't show what
23 the vendor authoring component would look like since it is
24 displayed from the point of view of the customer. On the other

1 hand, the message can be used to start a new message in the
2 "customer" component. There are different scenarios that could be
3 depicted by Figures 3-5.

4
5 According to the first scenario, Figure 3 shows a message
6 displayed in a customer component with the "customer" role. The
7 customer clicks on the "Click here to Automatically order...", and
8 the customer component starts a new message that is assigned the
9 role "vendor" (i.e. it will be received by a participant in the
10 vendor role). The new message with the "vendor" role is displayed
11 in the customer component is shown in Figure 4. This is an
12 interpretation that matches closely to the operation of the code
13 included in the parent application. However, in the original
14 code, the invocation of the new message is normally done by
15 hitting the "reply" button in the main interface rather than a
16 field displayed on the incoming message.

17
18 This functionality is useful if it is necessary for a message
19 author (e.g. a vendor) to control the assignment of role
20 information associated with various types of replies on an
21 individualized basis. For example, if it wanted certain types of
22 customers to have access to the online customer service and wanted
23 to encode this in the message body itself. In this case, some
24 customers would receive a message looking similar to the one

1 displayed in Figure 3 with an additional line that says "Click
2 here to request customer support" in which case a new message with
3 a role assigned to be "SalesInfo" would be created in the Sales
4 authoring/reading component.

5
6 According to the second scenario Figure 3 shows a message
7 displayed in a "controller" component that is capable of
8 responding to an input pair (message data + user action) by
9 opening new messages in other components. When the customer
10 clicks on the "Click here to automatically order..." the
11 "controller" component opens a new message in author mode (perhaps
12 tagged with the "customer" role) in the "customer" component.
13 This scenario could be implemented by sending an HTML-like message
14 with a message MIME type set up to identify the controller
15 component, e.g. "x-ControllerMsg" whose body consists of a set of
16 data pairs (display_string/associated component to open/ data
17 parameters for component). For example, the screen shown in
18 Figure 3 could be generated by a message that had a "Content-Type
19 = x-ControllerMsg" and a message body that consists of:

20
21 ("One Day Special.../nil/nil)

22 ("Click here to go to our Web Page for more details."/ text/HTML /

23 "http://www.sales.com")

1 ("Click Here to Automatically Order this Product by Email"/ x-Sales/Customer
2 /"PID 121121")

3
4 Existing mail clients are almost capable of implementing this sort
5 of behavior using HTML and Javascripts. If an existing mail
6 client were modified to be able to run Javascripts, the message
7 displayed in Figure 3 could consist of HTML and Javascripts. When
8 the user clicks on the "Click here to automatically..." A
9 Javascript could run that opens the customer component with a new
10 message with the vendor role (as shown in Figure 4). However, as
11 mentioned above, this type of implementation would open the mail
12 client user to malicious Javascripts. According to the preferred
13 embodiment of the invention, no executable code is contained in a
14 message, only tags which are read by the authoring/reading
15 component.

16
17 Figure 6 shows another similar example of a message handling
18 component/role is used in e-commerce. Figure 6 shows an email
19 message 60 which was generated by an "auctioneer" component (or an
20 authoring/reading component in auctioneer mode). Figure 6 shows
21 the message as it appears on the computer screen of a "bidder".
22 This message contains several graphic components 62, 64, three
23 text message fields 66, 68, 70, and an input template 74. The
24 message 60 is a combination incoming/outgoing template. The

1 bidder fills in the appropriate parts of the template 74 and
2 clicks on the "reply" button 76. The bidder's authoring/reading
3 component composes an appropriate message to the auctioneer and
4 sends it automatically and transparently. The user will see the
5 message 60 disappear and preferably see a dialog box or window
6 notice saying "Message Sent" or the like.

7
8 As mentioned throughout above, the concept of role may be
9 associated with a particular component or with a message created
10 by the component. In the auction example, it is desirable that
11 the non-auctioneer component have at least two modes of operation,
12 one being the role of bidder and the other being the role of
13 seller. The bidder role is illustrated with reference to Figure 6
14 described above. The seller role is illustrated with reference to
15 Figure 6a which shows a message template 61 which is created when
16 the user chooses to post a for sale message. The template 61
17 includes a title field 63, a category field 65, a description
18 field 67 and an option checklist 69.

19
20 According to a presently preferred embodiment, the concept of
21 role is associated with the message and authoring/reading
22 components are capable of creating and reading messages having
23 different roles. In the KidKode® embodiment of the invention, the
24 role of the message is contained within the body of the message.

1 According to other embodiments, the role of the message is encoded
2 in the MIME type, subtype, a parameter in the "Content_Type"
3 header, or in the subject header. The processing of a message
4 with a specified role may be carried out either by an
5 authoring/reading component designed specifically to handle
6 messages with the specified role or by an authoring/reading
7 component that is built to handle all messages of a specified MIME
8 type. When the authoring/reading component handles all messages
9 of a given MIME type it includes within it the logic necessary to
10 process messages with different roles differently.

11
12 As mentioned above, in the KidCode® embodiment, the message
13 role is encoded in the message body and is handled entirely by the
14 authoring/reading components. This has advantages over an
15 alternative that records the message role in a message header
16 field because the mail email client can be independent of any
17 knowledge of roles and does not need to be encumbered with role
18 handling logic that is best left to component designers. However,
19 this places a restriction on the design of authoring/reading
20 components. If role information is embedded in the message body,
21 it is not advisable to allow the creation of authoring/reading
22 components specialized to handle certain roles. To do so would
23 require that the main mail client software be capable of parsing
24 the message body in order to find the role information and,

1 thereby invoke the proper authoring/reading component for a given
2 message. Good software design principals require a strict
3 separation between the data available to the main mail client and
4 data that is processed exclusively by authoring/reading
5 components. Maintaining this separation is very important to
6 ensure integrity of the API which allows independent software
7 developers to build authoring/reading components that can be
8 guaranteed to work properly with a main email client program.

9
10 Nevertheless, there are many types of transactions in which
11 each participant in an email exchange acts in a single role. For
12 example, in the Doctor/Patient application described below, the
13 individual who is the patient normally does not take on the Doctor
14 role. In this case, it is both more secure and more efficient to
15 construct separate authoring/reading components to handle Doctor
16 or Patient messages. It is also desirable to recognize that these
17 are two sides of the same transaction and encode all messages
18 associated with the transaction with the same MIME type.

19
20 A solution that enables both the development of
21 authoring/reading components specialized to handle particular
22 roles and guarantees data separation as described above is to
23 encode role information as a subtype of the MIME type in the
24 "Content-type" header field. Using this method, the main email

1 client would properly invoke the correct authoring/reading
2 component in the case that an independent authoring/reading
3 component has been constructed for handling a specified message
4 role. Another alternative is to encode role information in the
5 parameter that may be optionally included in the "Content-type"
6 header as defined by the MIME standard. In this case, the main
7 email client maintains data separation with respect to the message
8 body but is required to parse the Content-Type parameter. The
9 additional complexity necessary in the main mail client software
10 causes this to be an inferior solution.

11
12 Figures 7 and 8 illustrate authoring/reading components
13 having the roles of "doctor" and "patient". Figure 7 illustrates
14 a message template 80 for use by a doctor. The message template
15 is divided into three main parts 82, 84, 86. The first part 82
16 presents the doctor with pop up menus for selecting, for example,
17 patient name, prescription number, test provider, test number,
18 etc. Radio buttons are also used to enter information about
19 prescription refills, generic substitution, etc. The second part
20 84 is a text editor window where the doctor may compose a personal
21 message to the patient. The third part 86 is a text entry
22 template HMO information, consultation charge, etc. After the
23 doctor fills out the message template 80 and sends it, the
24 authoring/reading component prepares a message to the patient and

1 a message to the HMO. Optionally, the authoring/reading component
2 could also prepare messages to a pharmacy, to a test lab, etc.

3
4 Figure 8 illustrates the message 90 received by the patient
5 as a result of the doctor filling in the template 80 (Figure 7).
6 The message 90 presents the role of patient and organizes
7 information in a manner useful to the patient. The message 90
8 includes a text field 92 which displays the text message composed
9 by the doctor and entered into the text field 84 in Figure 7 and
10 several check box fields 94, 96, 98. Field 94 allows the patient
11 to request information about a medical condition. Field 96 allows
12 the patient to select certain parameters regarding the filling of
13 a prescription and to request a copy of diagnostic test results.
14 Field 98 allows the patient to purchase and obtain information
15 about vitamins. Figure 8 also illustrates a pop-up dialog box 100
16 which alerts that patient regarding to the billing information
17 entered by the doctor in field 86 of Figure 7. When the patient
18 selects parameters regarding the filling of a prescription, the
19 patient component automatically sends email to the pharmacy with
20 only the information needed by the pharmacy to fill the
21 prescription.

22
23 From the foregoing, those skilled in the art will appreciate
24 that the concept of "roles" as implemented herein may be expanded

1 to a number of different useful applications of interactive email.
2 The roles described and illustrated above include customer,
3 vendor, database, doctor, and patient. Previously incorporated
4 parent application serial number 09/209,162 disclosed several
5 examples of teacher and student roles. Those skilled in the art
6 will appreciate that the principles describe herein can be applied
7 to roles such as service provider, technical support, attorney,
8 client, and organization members having different security
9 clearances. Service provider and technical support will present
10 templates requesting information needed to diagnose a problem.
11 Reply mail with data fill-in can be parsed by an artificially
12 intelligent mail reader and tentative diagnoses sent by email to a
13 human representative. The attorney and client roles may be
14 implemented in a manner similar to the doctor and patient roles.
15 The roles of organization members have different security
16 clearances can be implemented in a manner whereby a message
17 created by someone with the highest security clearance is
18 classified in different ways so that only those with the proper
19 clearance can read it. For example, it is possible to create a
20 message where some parts of the message are highly classified and
21 other parts of the message are given lower classifications.
22 Readers with high security clearance will be able to read all or
23 most of the message. Readers with low clearance will only be able
24 to read parts of the message.

1 As stated above, normally the role information associated
2 with each message is assigned to the message when it is created or
3 when it is sent. A role can often be automatically assigned to a
4 message by the authoring/reading component used to author the
5 message. The role given to a message may be based on many
6 factors including the state of a previous message (or sequence of
7 messages), information selected by a message author, and
8 information associated with the message recipient and/or the
9 message author. In the KidCode® embodiment, the authoring/reading
10 components built for children's educational games automatically
11 assign a role to a message on the basis of the state of a previous
12 message.

13
14 Figure 9 shows the state diagram used to determine how roles
15 are automatically assigned to KidCode® messages. In a preferred
16 embodiment of the KidCode® coder/decoder games, when a user starts
17 a new message at 110 by pressing the message authoring button in
18 the main client program, the message is tagged as a "new" message
19 and treated as a "coder" message. When the message is sent, the
20 message role is set to "decoder" at 112 because the message
21 recipient will play the role of decoder of the puzzle. When the
22 decoder sends the reply to the message the role of the outgoing
23 message is set to "coder" at 114. When the coder participant
24 receives that reply from the decoder, the fact that the message is

1 tagged with the "coder" role causes appropriate "coder" tools to
2 be made available by the authoring component. As is shown in
3 Figure 9, each time a reply message is created a copy of the
4 original message is made and the message role is set to the
5 opposite role.

6
7 Figure 10 elaborates on the manner in which messages are
8 displayed by the KidCode® Rebus and Grid message handling
9 components. Whereas Figure 9 describes transitions undergone by a
10 message (or message descendants), Figure 10 shows the transitions
11 of the state of an authoring/reading component. This Figure
12 provides a more complete picture of how an authoring/reading
13 component can use the message role and the message display mode (a
14 property of the authoring/reading component that normally takes
15 one of the two values, "display" or "author") to determine how a
16 message should be displayed and what tools are available to the
17 message author or reader. On start 116, the component has three
18 options. One option is to create a new message in the coder role
19 (component author mode) 118. Another option is to open a message
20 in the decoder role (component display mode) 120 in order to view
21 a puzzle created by another. Still another option is to open a
22 message in the coder role (component display mode) 122. This
23 option would be used to view a response by a decoder to a coded
24 message created by the current user. If the user chooses to reply

1 to this message, the message role is changed to decoder (component
2 author mode) 124.

3
4 The above illustration illustrates an important distinction
5 to note regarding the interaction of the message mode and the
6 message role. For many types of interactions it is desirable to
7 control the information that is seen by the user depending upon
8 the role that the user has in a given transaction or series of
9 exchanges. For example, messages that have already been sent and
10 reside in the outbox are read only messages (i.e. always opened
11 with the mode set to "display"). In the case of an authoring
12 component such as the KidCode® Rebus or Grid, in which the role of
13 a sent message is in an incorrect state for the message author
14 (i.e. a decoder's message that has been sent has the "coder"
15 role), a "display" mode message must be viewed as having a
16 different role than it actually carries. If a Rebus decoder were
17 to see his reply message after it has been sent with a "coder"
18 role, he would be able to view the answers to the puzzle in the
19 messages in his outbox. The logic to handle the various
20 interactions between message modes and message roles are included
21 in the authoring/reading component. In this way the main email
22 component is separated from all aspects of message roles which are
23 properly the domain of an particular messaging application.

24

004290" 8240960

1 There have been described and illustrated herein several
2 embodiments of electronic mail software with modular integrated
3 authoring/reading software components including methods and
4 apparatus for controlling the interactivity between mail authors
5 and recipients. While particular embodiments of the invention
6 have been described, it is not intended that the invention be
7 limited thereto, as it is intended that the invention be as broad
8 in scope as the art will allow and that the specification be read
9 likewise. Thus, while particular APIs have been disclosed, it
10 will be appreciated that other APIs could be utilized provided
11 that the basic functionality described herein is maintained.
12 Also, while "installable" components have been discussed in
13 detail, it will be recognized that permanently installed
14 components could be used with similar results obtained, less the
15 ability to upgrade functionality in an easy manner. Moreover,
16 while particular configurations have been disclosed in reference
17 to the relationship between the main email client, the message
18 handlers, and the mailbox handlers, it will be appreciated that
19 other configurations could be used as well provided that the basic
20 functionality of the invention is maintained. It will therefore
21 be appreciated by those skilled in the art that yet other
22 modifications could be made to the provided invention without
23 deviating from its spirit and scope as so claimed.

24